***Министерство образования Республики Беларусь***

***Учреждение образования***

***«Брестский государственный технический университет»***

***Кафедра ИИТ***

**Лабораторная работа №4**

**По дисциплине АОИС за III семестр**

**Тема: «Моделирование ассоциативная памяти**

**при помощи нейронных сетей»**

**Выполнил:**

Студент группы ИИ-15 (1)

2-го курса

Волк И. А.

**Проверил:**

Михно Е. В.

Брест 2018

Цель работы: Изучить обучение и функционирование релаксационных ИНС в качестве ассоциативной памяти при решении задач распознавания образов.

Задание:

1. Написать на любом ЯВУ программу моделирования ИНС для распознавания векторов согласно варианту. ИНС содержит n нейронных элементов в первом слое и m во втором слое. Если n меньше размерности вектора, тогда из вектора использовать только первые n элементов.
2. Провести исследование полученной модели. При этом на вход сети необходимо подавать искаженные образы, в которых инвертированы некоторые биты. Критерий эффективности процесса распознавания - максимальное кодовое расстояние (количество искаженных битов) между исходным и поданным образом.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вариант | n | m | № векторов | Модель ИНС |
| 17 | 15 | 4 | 4,11,8,7 | Хэмминга |

Код программы:

//main.cpp

#include <iostream>

#include <algorithm>

#include "hammnet.h"

#include "nnet.h"

#define endl std::cout << std::endl

#define out(a) std::for\_each(a.begin(), a.end(), [](int i) {std::cout << i << " "; })

#define proc(a) net.process(a, test\_result)

int main()

{

constexpr int N = 7;

constexpr int NUM\_OF\_SAMPLES = 4;

typedef Net::Sample<N> Sample;

typedef Net::SampleSet<N, NUM\_OF\_SAMPLES> SampleSet;

typedef HammingNetwork::HNet<N, NUM\_OF\_SAMPLES> Net;

Sample s4 = { 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0 };

Sample s11 = { 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1 };

Sample s8 = { 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1 };

Sample s7 = { 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1 };

SampleSet set = { s4, s11, s8, s7 };

Net net(set);

std::vector<int> test\_result(NUM\_OF\_SAMPLES);

proc(s4);

out(test\_result);

endl;

proc(s11);

out(test\_result);

endl;

proc(s8);

out(test\_result);

endl;

proc(s7);

out(test\_result);

endl;

Sample test = { 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0 };

proc(test);

out(test\_result);

endl;

std::cin.get();

}

//nnet.h

#pragma once

#include <iostream>

#include <vector>

namespace Net

{

template<int N>

class Sample

{

std::vector<int> sample;

public:

Sample()

: sample(N)

{

for (int i = 0; i < N; i++)

sample[i] = -1;

}

Sample(int sample[N])

: sample(N)

{

for (int i = 0; i < N; i++)

this->sample.push\_back(sample[i]);

}

Sample(std::initializer\_list<int> list)

: sample(N)

{

int i = 0;

for (int x : list)

if (i < N)

sample[i++] = (2\*x - 1);

for (; i < N; i++)

sample[i] = -1;

}

int& operator[](const int id)

{

try

{

if (id < 0)

throw 1;

if (id >= N)

throw 2;

return sample[id];

}

catch (int e)

{

std::cout << "int& HopfieldNetwork::Sample::operator[](const int id)\n"

<< "Error - " << e << std::endl;

system("pause");

exit(0);

}

}

bool operator==(const Sample<N>& other) const

{

for (int i = 0; i < N; i++)

if (sample[i] != other.sample[i])

return false;

return true;

}

Sample& operator=(const Sample<N>& other)

{

for (int i = 0; i < N; i++)

this->sample[i] = other.sample[i];

return (\*this);

}

int size() const

{

return N;

}

void view(const std::string& message = "\0", const int tab = 0)

{

if (message != "\0")

std::cout << message.c\_str() << ": ";

for (int i = 0; i < tab; i++)

std::cout << "\t";

for (int i = 0; i < N; i++)

std::cout << sample[i] << " ";

std::cout << std::endl;

}

};

template<int N, int SIZE\_OF\_SET>

class SampleSet

{

std::vector<Sample<N>> samples;

public:

SampleSet(Sample<N> samples[SIZE\_OF\_SET])

: samples(SIZE\_OF\_SET)

{

for (int i = 0; i < SIZE\_OF\_SET; i++)

for (int j = 0; j < N; j++)

{

this->samples[i][j] = samples[i][j];

}

}

SampleSet(std::initializer\_list<Sample<N>> list)

: samples(SIZE\_OF\_SET)

{

int i = 0;

for (Sample<N> x : list)

if (i < SIZE\_OF\_SET)

samples[i++] = x;

for (; i < SIZE\_OF\_SET; i++)

samples[i] = Sample<N>();

}

Sample<N>& operator[](const int id)

{

try

{

if (id < 0)

throw 1;

if (id >= SIZE\_OF\_SET)

throw 2;

return samples[id];

}

catch (int e)

{

std::cout << "Sample& HopfieldNetwork::SampleSet::operator[](const int id)\n"

<< "Error - " << e << std::endl;

system("pause");

exit(0);

}

}

};

}

//hammnet.h

#pragma once

#include "nnet.h"

namespace HammingNetwork

{

template<int N, int SIZE\_OF\_SET>

class HNet

{

const double e;

double W[N][SIZE\_OF\_SET];

int numOfNeurones;

double T[SIZE\_OF\_SET];

double V[SIZE\_OF\_SET][SIZE\_OF\_SET];

public:

HNet(Net::SampleSet<N, SIZE\_OF\_SET> samples)

: e(1.0/N - 1.0/(10.0\*N)), numOfNeurones(N)

{

for(int j = 0; j < SIZE\_OF\_SET; j++)

{

for(int i = 0; i < numOfNeurones; i++)

W[i][j] = samples[j][i] / 2.0;

T[j] = - N/2.0;

}

for(int j = 0; j < SIZE\_OF\_SET; j++)

for(int k = 0; k < SIZE\_OF\_SET; k++)

V[j][k] = (j == k)?(1):(-e);

}

double F(double S)

{

if(S > 0)

return S;

else

return 0;

}

void process(Net::Sample<N>& sample, std::vector<int>& result)

{

double P[SIZE\_OF\_SET];

for(int j = 0; j < SIZE\_OF\_SET; j++)

{

P[j] = 0;

for(int i = 0; i < N; i++)

P[j]+=W[i][j]\*sample[i];

P[j] -= T[j];

}

double Z[SIZE\_OF\_SET];

for(int k = 0; k < SIZE\_OF\_SET; k++)

{

Z[k] = 0;

for(int j = 0; j < SIZE\_OF\_SET; j++)

Z[k] += V[j][k]\*P[j];

}

double S[SIZE\_OF\_SET];

while(1)

{

for(int i = 0; i < SIZE\_OF\_SET; i++)

S[i] = Z[i];

for(int k = 0; k < SIZE\_OF\_SET; k++)

{

for(int j = 0; j < SIZE\_OF\_SET; j++)

Z[k]+=V[j][k]\*S[j];

Z[k] = F(Z[k]);

}

bool isDif = false; // is last Z different to current Z

for(int i = 0; i < SIZE\_OF\_SET; i++)

if(S[i] != Z[i])

{

isDif = true;

break;

}

if(!isDif)

{

for(int i = 0; i < SIZE\_OF\_SET; i++)

result[i] = (Z[i]!=0)?(1):(0);

return;

}

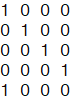
}

}

};

}

Результат выполнения:



Вывод: изучил обучение и функционирование релаксационных ИНС в качестве ассоциативной памяти при решении задач распознавания образов.